SAVANT 技术说明书

中国物品编码中心

2003 年 11 月 15 日

目 录

第一节 前言 3

1.1 解读器 3
1.2 SAVANT 3
1.3 EPC 信息服务 3
1.3 ONS－对象名解析服务 3

第二节 简介 4

2.1 EDGE SAVANT 5
2.2 INTERNAL SAVANTS 5
2.3 数据管理 5

第三节 事件管理系统 6

3.1 需求 7
3.2 系统结构 8
3.2.1 解读器接口 8
3.2.2 解读适配器 8
3.2.3 事件队列 8
3.2.4 事件过滤器 9
3.2.5 事件记录器 10

第四节 实时内存事件数据库 10

4.1 需求 10
4.2 系统结构 11

第五节 任务管理系统 12

5.1 需求 13
5.2 系统结构 13
5.3.1 任务管理器 14
5.3.2 SOAP 接口 14
5.3.3 类服务器 14
5.3.4 数据库 14

第六节 结论 15

参考文献 16

第一节 前言

无线射频识别（RFID）是一种用以识别、跟踪和定位物品的技术。推出这项技术的 Auto－ID Center 的梦想是建立一个对单品的通用唯一标识符。这个唯一的号码叫做 EPC（electronic product code），被封装在比较便宜的 RFID 标签里。EPC 网络也可以捕获或者通过互联网和授权得到产品的其他信息。

下面对 EPC 网络架构的各部分进行概要说明。

1.1 解读器

解读器负责探测进入解读范围的标签，也可以询问绑定或嵌入标签的其他传感器信息。Auto－ID 解读器说明书 1.0 定义了解读器与 Savant 和其他后台程序通讯的标准协议。Savant 也包含一个“适配器”，用以对不支持 Auto－ID 协议的旧的解读器提供接口。

1.2 Savant

Savant 是一个“中间件”，用来处理从一个或多个解读器发出的标签流或传感器数据（事件数据）。Savant 在向企业应用程序发送数据之前，要压缩数据量，对标签数据进行过滤、集成和计算。

如果现在采用条码标识的物品都用 Auto－ID center 的 EPC 标识，EPC 的网络管理软件就要以每秒几百万的速度处理事件。为了管理这个巨大的事件流，Auto－ID center 推出了分层、模块化的组件，就是 Savant。

1.3 EPC 信息服务

EPC 信息服务使请求服务可以得到以 PML 格式描述的关于 EPC 网络的数据。可以通过 EPC 信息服务发送的数据包括 Savant 收集的标签解读数据（例如：用于产品跟踪的序列号）、产品级数据（例如：产品信息、保质期等等）、物品种类级数据（例如：产品目录信息）。为了满足要求，EPC 信息服务将企业中存在的一系列数据源翻译成 PML 格式。如果 EPC 数据分配在供应链中，行业中可能形成 EPC 访问注册机，作为 EPC 信息服务接口描述的知识库。

1.3 ONS－对象名解析服务

对象名解析服务提供全球查询服务，将一个 EPC 翻译成一个或多个 URL，以方便找到进一步的产品信息。这些翻译得来的 URL 经常用来识别 EPC 信息，尽管通过 ONS 也可以将 EPC 与 Web 站点或者其他网络资源联系起来。

ONS 提供两种服务，静态的和动态的。静态 ONS 典型的应用是为产品制造

商维护的信息提供 URL，动态 ONS 服务负责管理运行在供应链上的产品信息。ONS 的运行机制与 DNS（域名解析服务）相似。

第二节 简介

不同应用程序对 EPC 处理的需求大相径庭。而且 EPC 仍然处在发展的初期，随着它的不断成熟，会对应用程序进行各种改进和变动。Savant 被定义成具有一系列特定属性的“程序模块”或“服务”，并被用户集成以满足他们的特定需求。这些模块设计的初衷是能够支持不同群体对模块的扩展，而不是能满足所有应用的简单的集成化电路。

Savant 是连接标签解读器和企业应用程序的纽带，代表应用程序提供一系列计算功能，在将数据送往企业应用程序之前，它要对标签数据进行过滤、总计和计数，压缩数据容量。为了减少网络流量，Savant 也许只向上层转发它感兴趣的某些事件或事件摘要。

Savant 程序模块可以由 Auto－ID 标准委员会定义，或者用户和第三方生产商来定义。Auto－ID 标准委员会定义的模块叫做标准程序模块。其中一些标准模块需要应用在 Savant 的所有应用实例中，这种模块叫做必需标准程序模块；其他一些可以根据用户定义包含或者排除于一些具体实例中，这些就叫做可选标准程序模块。本说明书中介绍的三个模块：事件管理系统（EMS）、实时内存数据结构（RIED）任务管理系统（TMS），都是必需标准程序模块，我们着重介绍这三个模块的设计需求和系统结构及实现细节。

Savant 具有树型等级结构，这种结构可以简化管理，提高系统运行效率。如图 1 描述了一个典型 Savant 系统结构。树型结构的叶节点叫做“Edge Savant”（ES），树的分支节点叫做“Internal Savants”（IS）。Savant 软件是处理从一个或多个解读器获得的标签或传感器数据（事件数据）流的“中间件”，是一种企业通用的管理 EPC 数据的架构，安装在等级式、分布式的服务器上。它可以安装在商店、本地配送中心，区域甚至全国数据中心中，具有数据捕获、监控、传送功能。下面两小节介绍这些 Savant 软件的分类。最后 2.3 节介绍 Savant 网络中的 EPC 数据管理。

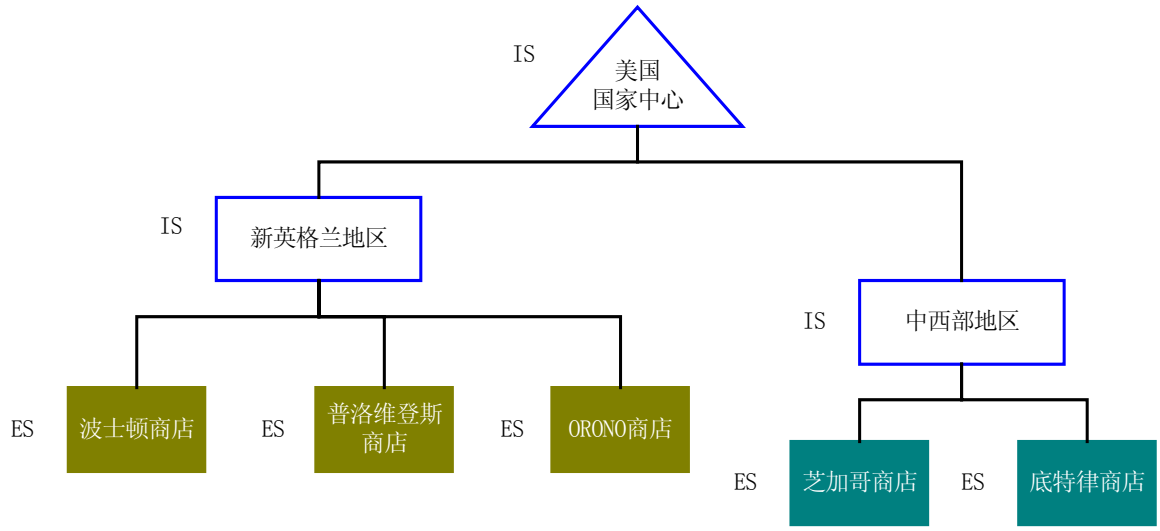

图 1 Savant 网络结构

2.1 Edge Savant

ES 是采集实时 EPC 数据的 Savant。典型情况下，Savants 软件装在商店、仓库、制造车间，甚至卡车上。ES 由它们在网络中的逻辑位置而得名：EPC 数据只有通过它们才能进入系统。为了以后恢复方便，ES 不停的捕获、监视、存储数据。在 Savants 等级结构中，ES 总是结构树的叶节点。

Edge Savant 与 RFID 的解读器相连。解读器不停的从标签中采集 EPC 数据，并向 Savant 传输。每次识读，Savant 都要保存一些信息，例如：标签的 EPC 码；扫描标签的解读器的 EPC 码；识读时间；以及与 EPC 不相关的一些信息，如解读器观测到的温度和地理位置。

2.2 Internal Savants

在 Savants 的逻辑等级中 Internal Savant 指内部节点，是 edge savants 的父节点或上级，IS 从他的下属 Edge Savant 中采集 EPC 数据。通常，IS 安装在企业的区域或国家级数据中心。IS 系统除从它的下级采集生数据外，还负责合计 EPC 数据。

2.3 数据管理

在 Savant 的网络结构中，尽管共同承担处理和维护系统数据的责任，但并不是所有的 Savant 都具有同等的加工、存储和接口要求。

既然我们要对不同等级的 Savant 分配数据，就必须解决以下问题：

（1） 在 Savant 网络结构中数据如何分布。

（2） 数据如何存取和查询。

（3） 数据如何维护。

我们可以采用多种方式在 Savant 中采集数据。通常，我们选择一个属性然后根据属性值采集数据。在我们的构架中，优先采用按照二维（或属性）分割数据的原则：时间和空间。因为 EPC 数据只能通过 Edge Savant 采集，必须在 Edge Savant 的上级（internal savant）中传输和分流。所以随着数据采集的不断进行，我们希望数据只通过特定的 Savant 收集。当按照空间属性分配数据时，我们规定从某个 Savant 采集的数据只能出现在它通向根节点的 Savant 中。换句话说，不管是不是直接上下级关系，一个内部 Savant 只能保存关于它下级 Savant 的数据。

图 2 描述了数据按照空间和时间分配的简单例子。数据在 savant 中按照时间垂直分配，同时每个父节点保存它的子节点的数据而不是兄弟节点的数据。

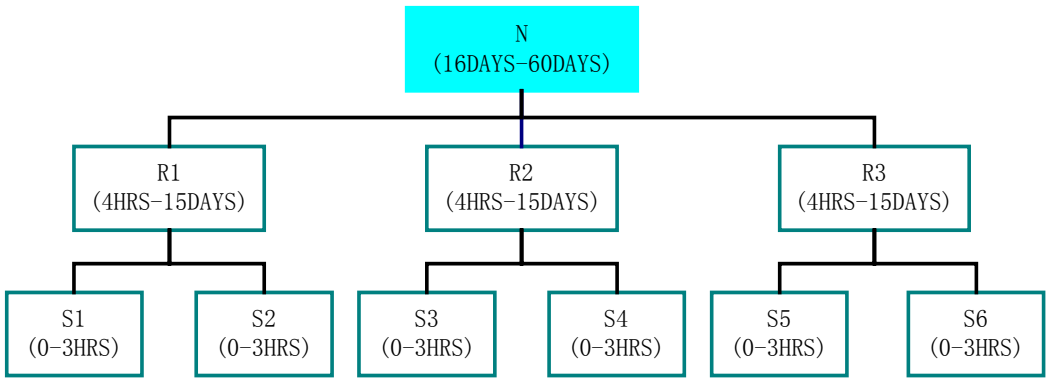

图 2 数据的典型分配

为了实现这种分配形式，除了从 Edge Savant 中采集的数据外，我们还需要保存额外的数据。对每个从 leaf savant 加载（或更新）的数据我们都要保存两个属性：Savant 的 ID 号和更新时间记录。图 3 是一个简单记录的图示。

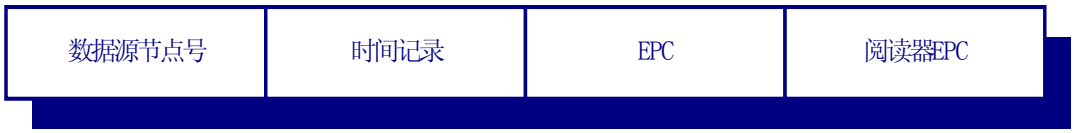

图 3 Savant 记录的结构

据此，只在网络的垂直方向上有数据移动。就是说，随着时间的推移，新数据采集上来，现有数据变成旧数据，它们系统的沿着网络树向上传输。在向上传输之前，EPC 数据在某个 Savant 中保留多长时间可以由用户配置。

所有 Edge Savant 都从网络中与它直接相连的解读器上采集数据。这些数据是解读器扫描获得的 EPC 的基础集合。最简单的情况下，集合只是所有 EPC 的列表；如果要求描述的比较详细，数据集合也许是一些 EPC 的前缀。另一方面，比叶节点高一级的内部 savant（internal savant）从子 Edge Savant 中采集数据并保存信息的集合。例如，图 3 中所示的新英格兰地区的地区中心可以保存从 Boston、Providence、Orono Savant 采集来的 EPC 信息的集合。

第三节 事件管理系统

这节讲述 Savant 的事件管理系统（Event Management System，EMS）。事件管理系统应用在 Edge Savant（ES）上采集标签解读事件，它与解读器应用程序通讯，管理解读器发送的事件流。在 Savant 系统中，EMS 是最重要的组件，它为用户提供了集成其他应用系统的平台，它的职责包括：

（1）允许不同种类的解读器写入适配器；

（2）以标准格式从解读器采集 EPC 数据；

（3）允许设置过滤器，以平滑或清除 EPC 数据；

（4）允许写各种记录文件，例如记录 EPC 数据存储到数据库中的数据库日志，记录 EPC 数据广播到远程服务器事件的 HTTP/JMS/SOAP 网络日志。

（5）对记录器、过滤器和适配器进行事件缓冲，使它们在不相互妨碍的情况下运行。

3.1 需求

EMS 帮助 Edge Savant 采集、缓冲、平滑和组织从标签解读器获得的信息。下面介绍对事件管理系统的性能要求。

需求 1：

标签解读器每秒可以上传数百个事件。每个事件都能够在处理 Savant 请求的基础上被恰当的缓冲、过滤和记录。因此，EMS 必须是一个高性能的系统。

需求 2：

不同种类的解读器工作在不同的协议下，EMS 必须支持多种解读器协议，因此 EMS 必须支持采用不同 Auto－ID Center 协议的 EPC 解读器。

需求 3：

EMS 读取的事件能够在 Savant 满足要求的基础上被过滤。下面几个例子描述对过滤器的要求：

（1） 平滑：有时，解读器会读错或丢失标签。如果 EPC 被读错，叫做积极识读错误，如果覆盖区内的 EPC 被漏读，叫做消极识读错误。平滑算法要清除那些被怀疑有积极或消极错误的识读。

（2） 协调：当多个解读器相互离的很近时，它们可能读到相同的 EPC。如果同一个 EPC 被不同的解读器上传两次，Savant 的流程逻辑会产生错误。协同过滤器会清除 EPC“不属于”的那个解读器的阅读。协同工作可以采用不同的运算规则。如果刚刚的几毫秒中，一个解读事件涉及不同的解读器识读同一个 EPC，协同运算法则就可以删除这一事件。如果当前解读器离标签比它应该“属于”的解读器离标签近，附加的逻辑应该允许当前解读器的事件通过。

（3） 转发：一个事件转发器应该有一个或多个输出。根据事件类型的不同，转发器可将事件传送为一个或多个输出。例如，事件转发器可以选择只转发解读器上传的非 EPC 阅读事件，如阅读时的温度。

因此 EMS 应该支持具有一个输入事件流，一个或多个输出事件流的“事件过滤器”。

需求 4：

经过采集和平滑的事件最终会被恰当的处理（记录）。事件的记录方式有多种，可以是：

（1）保存在像数据库这样的永久存储器中；

（2）保存在存储数据结构中，例如实时内存事件数据库（RIED）；

（3）通过 HTTP、JMS 或 SOAP 协议广播到远程服务器；

因此 EMS 应该支持多种“事件记录器”。

需求 5：

上面提到的这些具有采集、过滤和记录功能的“程序模块”，应该工作在独立的线程中，不相互妨碍。这可以使 Savant 控制在处理解读器输入流时出现的“spkies”。因此 EMS 应该在不同的线程中启动处理单元，而且能够在单元间缓冲事件流。

需求 6：

最后，EMS 应该能够实例化和连接上面提到的事件处理单元。这些加工单元最普遍的组织形式是“有向图”，图的节点就是加工单元，图的边就是事件流。

而且，因为事件不会形成循环，我们可以把图结构简化，看成“非循环有向图”（directed acyclic graphs ,DAGs）。在 DAG 中远程机器可以获得事件流中的一些事件。因此，我们要求 EMS 应该能够在允许处理单元任意构造的基础上，将它们实例化为非循环有向图。

需求 7：

EMS 应该允许远方机器登录和注销到动态事件流中。

3.2 系统结构

这一节介绍 Savant 事件管理系统的结构。图 3 描述了事件管理系统（EMS）的设计及与其他应用的接口。EMS 有以下组成部分：

3.2.1 解读器接口

解读适配器直接或间接的与解读器进行通讯，并收集解读器探测到的事件。然后将这些事件写入解读器接口。解读器只负责识读和传输现有事件，不具备判断和记忆功能。例如，解读器可以标识识读范围内所有的标签，但无法判断是否读错或漏读标签。只要识读区域内的标签不变，解读器就持续发送事件流，标识它发现的所有标签。事件记录中除了标识 EPC 和 EPC 的读取时间外，还要生成非 EPC 信息（如温度、湿度等）和状态事件（告诉 Savant 系统，一个简单的解读器处理过程已经完成）。

图 4 描述多个解读器采用不同的协议向事件记录器发送信息的方式。

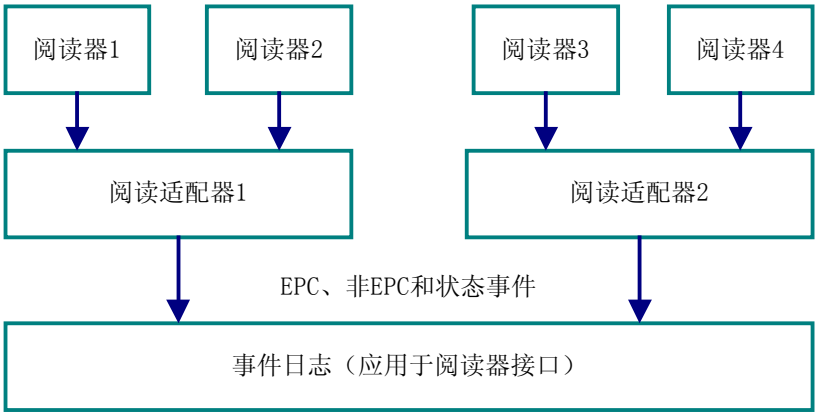

图 4 解读器接口应用

3.2.2 解读适配器

解读适配器与解读器通讯，采集 EPC 事件。它是事件制造者，向任何有解读器接口的“事件消费者”传输解读器事件。解读器可以采用多种物理方式与计算机进行通讯（如以太网、串口线），也许采用不同的通讯协议，为了支持不同种类的解读器，Savant 提供多种解读适配器接口，其中包含登录解读器和获得事件的具体执行细节。在解读适配器中，必须把事件分成 EPC 事件、非 EPC 事件、状态事件，以使它们在 Savant 系统中被恰当的处理。

3.2.3 事件队列

事件队列是一个异步排队系统，但采用同步机制处理多种解读器事件记录。排队系统可以记录被多种解读适配器读出的事件，并将它们传输到系统中注册的

所有解读器事件记录器中。当事件被解读器制造，解读适配器消化后，就被发送到事件队列中。在事件队列中，事件被发送到过滤器或记录器中，根据过滤器定义的不同，不同的事件被过滤或记录下来。事件记录器可以在事件队列中实时的登录或注销。

排队系统采用多进程增加系统的吞吐量。例如，一个数据库事件记录器，可能把绝大多数时间用在磁盘读和写上，但不会减缓向远程服务器传输实时事件的网络事件记录器的速度。因为事件队列既不是事件制造者又不是事件消费者，它们被叫做事件传输者。

图 5 描述了异步事件队列系统。在输入流的指针处，来自输入队列的事件被加到循环队列中，当到达队列末尾时，指针返回队列头。

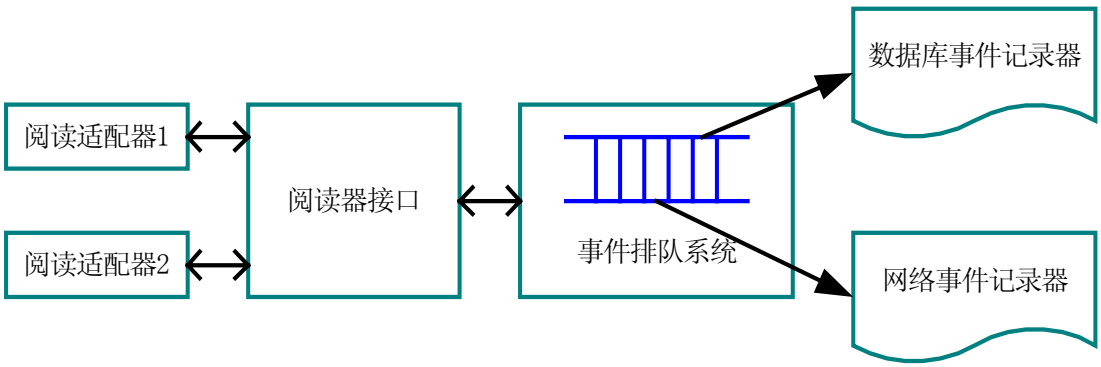

图 5 事件队列描述

3.2.4 事件过滤器

事件过滤器处理输入事件流，并向一个或多个输出事件流传输事件。它对事件进行分类，只向其他过滤器、记录器和应用程序发送它们感兴趣的事件。过滤器可以像拼装玩具一样被一个接一个的拼起来，以获得期望的事件。例如一个平滑过滤器可以和一个产品过滤器结合，将反盗窃应用程序感兴趣的事件分离出来。

可用的过滤器包括很多种，例如三个典型的过滤器：产品过滤器、时间过滤器和平滑过滤器。产品过滤器可以只发送与某一产品或制造商相关的产品信息，就是说，过滤器只发送某一范围或方式的 EPC 数据。时间过滤器可以根据时间记录来过滤事件，例如，一个时间过滤器可能只发送最近 10 分钟内的事件。平滑过滤器负责处理那些出错的情况，包括漏读和读错。所以在判断一个标签是否离开识读区域以前，解读器发送的数据必须经过过滤器处理，否则无法判断本次识读是不是出错。当解读器发送某一数量的事件都标识某个标签识读失败时，可以认定标签已经超出识读范围。例如，如果我们设定识读失败的极限数是 5，当连续 5 个事件标识解读器无法识读某个标签时，我们认定这个标签已经超出识读区域。

与事件队列不同，事件过滤器通常是同步应用。事件过滤器可以被放在事件制造者和事件消费者间，对事件进行平滑、协调和转发。因为事件队列既不是事件制造者也不是消费者，它们被叫做事件转发者。图 6 描述了事件队列。

图 6 事件过滤器

3.2.5 事件记录器

除了不向其他过滤器和记录器发送事件外，记录器与过滤器在本质上是相同的。事件记录器支持解读器接口的多种运行方式，允许对事件进行各种处理。例如，解读器接口的一个应用可以将信息储存在数据库中，另一个应用则将信息放在存储数据结构中，同时另一个应用可以将事件通过 HTTP、JMS 或 SOAP 协议广播到远方服务器。事件记录器也叫做“事件消费者”，因为它们消费事件流中的输入事件。

以上处理单元在 EMS 系统启动时被组织成 DAG，只有它的所有附件都启动后，EMS 才能启动某个处理单元。EMS 的配置文件定义了这个 DAG。启动时，“EMS 构造解析器”加载这个文件，然后按照文件说明启动处理单元。

第四节 实时内存事件数据库

这一节介绍 Savant 实时内存事件数据库（RIED）。RIED 是一个用来保存 Edge Savant 信息的内存数据库。

Edge Savant 保存和组织解读器发送的事件。事件管理系统（EMS）提供过滤和记录事件的框架，记录器可以将事件保存在数据库中。但是，数据库不能在一秒内处理几百次以上的交易。RIED 提供了与数据库一样的接口，但性能要好的多。

应用程序可以通过 JDBC 或本地 Java 接口访问 RIED。RIED 支持像 select,update,insert 和 delete 这样的 SQL 操作，还支持一部分 SQL92 中定义的数据操作方法。RIED 也可以保存不同时间点上数据库的“快照”。保存过期快照是为了满足 4.1 节讨论一些应用的要求。

4.1 需求

在本节中我们将看到为什么 RIED 系统是一个简单、高性能、多版本内存数据库。

让我们先看看解读器读取的 EPC 的保存方法。这看起来好像一个微不足道的

应用，但是让我们看看这个应用提出的性能要求。假设扫描到 10000 个物品，解读器每秒都要发送扫描报告，内存数据库每秒就要处理 10000 次交易。这个数据是比较保守的估计：我们的模型要求有效的规划以满足处理大容量解读的要求。因此我们只能存储解读器新读取的 EPC。因此，我们有：

需求 1：RIED 应该是一个高性能的内存数据库。

例如保存监视器的过期快照是为了满足偷窃监视和备份的要求。RIED 可以为过期信息保存多个只读快照。例如，数据库中可以保存监视器的两个过期快照，一个是一天的开始，另一个是每一分钟的开头。现有的内存数据库系统不支持对永久信息的有效管理，所以我们有：

需求 2：RIED 应该是一个多版本的数据库，即能够保存多种快照的数据库。

下一节我们介绍满足上述要求的 RIED 系统的结构。

4.2 系统结构

这一节介绍 Savant 实时内存事件数据库的结构，我们先看看 RIED 设计中的折衷之处，然后介绍 RIED 中用到的各种组件和数据结构。为了达到上述要求，在 RIED 设计过程中不得不做出一些让步。

1．降低 DML 的复杂程度

基于硬盘（disk-based）的 DBMS 携带大量逻辑以保证与积攒了 20 年之久的 SQL 属性兼容。较新的标准对其中的许多属性提出质疑，我们的模型尽量避免这些额外负担。

2．降低 DDL 的复杂程度

典型的 RDBMS 系统允许运行期间修改数据库。为了简化 RIED 数据结构的设计，数据库运行时 RIED 不允许进行 ALTER 操作。取而代之的是，启动时 RIED 从 DDL 文件中加载表结构。

3．仅对简单连接的有效支持

基于索引的 B 树支持基于“<”“>”的有效搜索和连接操作，而 RIED 仅支持“＝”查询和简单的连接。

4．最优化简单查询

典型数据库有复杂的查询最优化程序。在查询优化中决定链接的顺序是最重要的问题。一些数据库，例如 PostgreSQL，用遗传算法优化查询。可是 RIED 希望用户直接在 From 查询语句中决定连接顺序。

5．无约束维护和触发器 No constraint maintenance and triggers

一些数据库操作的时间花费在维护约束条件上。一些约束例如外来关键字和列检查，每次更新操作时都要评价。为了满足高效和简单的要求，RIED 不支持约束维护或触发器。

RIED 维护从解读器传来数据的各种状态快照。如果数据库当前的快照标记为 S0，以前的状态快照就被标记为 S1，S2……Sn。

对数据库的每个改变（更新操作例如：insert ,delete 或 update），都涉及序列号。每个快照 Si 都与一个序列号 Ti 关联。表示快照 Si 中保存 Ti 更新操作后的数据库状态。

RIED 用各种数据结构保存数据。各种数据结构都能在最新的快照执行更新操作并查询先前所有的快照集合，存储模型包含以下组件：

1．JDBC 接口：

JDBC 接口提供 RIED 表的 JDBC 外观。这个行业标准接口允许远程机器访问数据库，并允许使用标准 SQL 查询和用标准 URL 语言定位数据库。

2. DML 解析器

DML 解析器用来解析 SQL 数据操作语言。它包含标准 SQL select，insert 和 update 命令。DML 解析器只包含全部 SQL92DML 规定的一部分。而且也不支持 SQL92 中没有规定的操作。

3. 查询优化器

查询优化器处理 DML 解析器的输出，用保证存储模型能够被查询到的方法将它转换成执行计划。表定义中定义的查询路径被用来查找有效的执行计划。

4. 本地查询处理器

本地查询接口处理直接由应用程序或 SQL 解析器发送的执行计划。

5. 分类数组

本地队列处理器进行排序、集合和链接操作时用到分类数组。分类数组用到哈西表（当链接和聚合时）和有效分类算法（当排序操作时）。

6. 数据结构

RIED 应用（efficient thread-safe persistent data structures）有效线程安全永久数据结构保存表的不同快照。这些永久数据结构允许持续创建新的快照，它们在实时操作中也是必须的。一些 RIED 数据结构要求远程过期信息的碎片帐集。序列管理器保存当前序列号的线程安全值。

7. DDL 解析器

DDL 解析器处理表定义文件，并且在存储模型中初始化不同的数据结构。DDL 解析器也负责在 DDL 中定义的搜索路径的查询。

8．回滚缓冲器

RIED 中执行的操作可以提交或回滚，回滚缓冲器中存储所有对数据库的操作直到它们被提交。

第五节 任务管理系统

这一节我们介绍任务管理系统（TMS）。Savant 软件用用户定制的任务进行数据管理、数据监控，TMS 就负责管理由上级 Savant 或企业应用程序发送到本级 Savant 的任务。一般情况下，一个任务可以等价于多任务系统中的进程。Savant TMS 管理任务类似于操作系统管理进程。任务管理系统具有许多一般线程管理器和多进程操作系统不具有的特点，例如：

- ✧ 任务进度表的外部接口；
- ✧ 独立的（Java）虚拟机平台，包含从冗余类服务器（class server）中随选（on-demend）加载的统一库；
- ✧ 用来维护永久任务信息的健壮的进度表，和在 Savant 碎片或任务碎片中重启任务的能力。

Savant TMS 使分布式 Savant 的维护变得简单。企业可以仅仅通过在一组类服务器保存最新任务和在 Savant 中恰当的安排任务进度来维护 Savant。然而，硬件和核心软件，例如操作系统和 Java 虚拟机，必须定期升级。

写入 TMS 的任务可以获得 Savant 的所有便利条件。TMS 可以完成企业的多

种操作，例如：

- ✧ 数据交互，即向其他 Savant 中发送产品信息或从其他 Savant 获取产品信息。
- ✧ PML 查询：查询 ONS/PML 服务器获得产品实例的稳态或动态信息。
- ✧ 删除任务进度：确定和删除其他 Savant 上的任务。
- ✧ 值班报警：当某些事件发生时，警告值班人员，例如需要向货架补货、怀疑小偷和产品到期。
- ✧ 远程上载：向远处的供应链管理服务器发送产品信息。

5.1 需求

这一节介绍 Savant 任务管理系统的各种需求。我们可以看到为什么任务管理系统应该是一个有较小存储注脚，建立在开放、独立平台标准上的健壮的系统。

Savant 将要成为一个具有分布式产品管理框架的编译模块。在第二节讲述的 Savant 的等级结构，为任务管理器提出了一些要求。就像前面提到的，并不是所有等级式网络结构中的 Savant 软件都具有相同的处理、存储和接口要求。例如，内部 Savants 要求有较高的存储和处理能力。而 Edge Savant 则必须通过接口与解读器和警告消息服务器进行通讯。

需求 1：

Savant TMS 应该是具有较小存储处理能力的独立的系统平台。不同的 Savant 应该选择不同的工作平台。一些工作平台，尤其那些需求量很大的 Savant 所工作的平台，可以是进行低级存储和处理的便宜的嵌入式系统。

需求 2：

对网络上所有 Savant 软件进行定期升级是一项艰巨的任务，如果 Savant 基于简单维护的原则对代码进行自动升级则是比较理想的。所以我们要求 Savant TMS 能够对执行的任务进行自动升级。

需求 3：

Savant 需要为任务时序提供外部接口，为了满足公开和协同工作的系统要求，Savant TMS 应该为任务的时间进度、监控和删除提供定义完美的、可协同工作的外部接口。

需求 4：

为了将 TMS 设计从任务设计中分离出来，需要在一个独立的语言平台上，用简单、定义完美的软件开发工具包（SDK）描述任务。

5.2 系统结构

这一节介绍 Savant TMS 结构，TMS 中有些特殊要求以满足上述对系统的要求。Savant 结构基于开放的标准和技术，例如：Java，XML（Extensible Markup Language）和 SOAP（Simple Object Access Protocol）。图 7 描述了 Savant TMS 的系统结构。

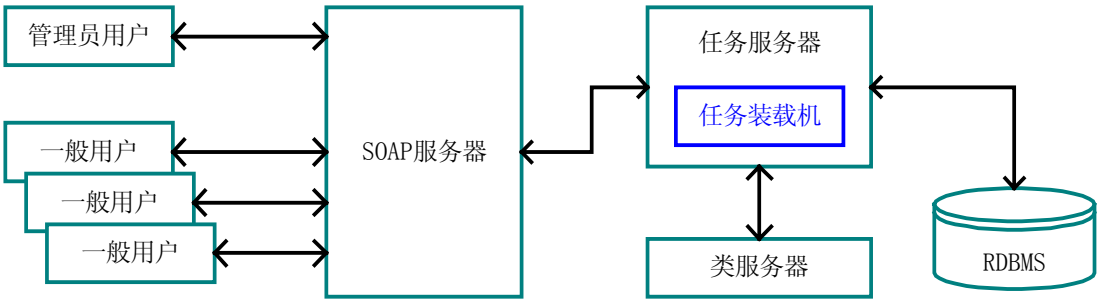

图 7 TMS 描述

系统包含的组件有：任务管理器、SOAP 接口、类服务器、数据库

5.2.1 任务管理器

任务管理器负责运行和维护 Savant 中的任务，这些任务代表用户在 Savant 端运行。所有的任务都从属于系统中的时间表，这个表决定任务多长时间运行一次，是否应该继续运行等等。依靠这个时间表，任务管理器才能决定每个时间步中哪个任务应该执行。下面介绍如何满足任务管理器的要求，根据任务类型和任务相关的时间表安排，任务管理器如下动作：

1．一次性任务

如果请求是一次性查询，任务管理器就处理查询任务并返回结果。

2．循环任务

如果请求中含有“循环时间表”，任务管理器将时间表存入“永久存储区”，然后按照时间表规定执行任务。

3．永久任务

如果请求是持续执行的永久任务，任务管理器定期监视任务。当有错误出现时，任务管理器只是重新启动该任务。

5.2.2 SOAP 接口

SOAP 服务器负责表明任务管理器的功能和接口，它使任务管理器成为可以被所有系统统一访问的 SOAP 服务。通过提交包含任务管理器表述详细办法的简单配置描述符文档，可以获得接口配置，随着任务管理器的成熟和它的 API 的扩展，可以通过增加新的表述办法来升级描述符。

5.2.3 类服务器

为了使系统任务可以动态添加，一旦新类可用，任务管理器就指向类服务器下载关于需求的新类。这可以在系统中很容易实现升级、新任务添加和修改，而无需重新启动。

5.2.4 数据库

数据库为任务管理器提供永久存储区。数据库中保存了提交的任务和任务时间表的细节。因此，所有向系统提交的任务都会留下任务管理器的碎片。每次循环，任务管理器都查询任务数据库并据此更新相关记录。

任务管理系统提供两个界面：

1．管理界面

允许远程管理和监控任务管理器的一组页面，通过这个界面，系统管理员可以进行如下操作：

♦ 启动、停止任务管理器

- 增加、删除、查看现有任务

2．系统界面

这个界面允许第三方实体通过 SOAP 消息与系统交互。每个希望与系统通讯的外部实体都必须登录到适配器或服务器，适配器或服务器处理进入的请求并把它们转发到任务管理器。这些传输都是通过 SOAP 消息进行通讯的。

下一节，我们介绍上述组件的应用细节，包括 TMS 提供的 SDK。

第六节 结论

Savant 是用以连接标签解读器和企业应用程序的“中间件”，Savant 软件与大多数企业软件的不同之处在于它不是一个集中式的应用程序，而是具有分布式结构的分层管理软件。Savant 软件可以安装在商店、配送中心、地区销售中心，生产厂家，甚至在卡车和货运飞机上。每一层次的 Savant 软件都能够收集、存储、整理信息并与其他 Savant 软件互动。例如：商店 Savant 能够通知配送中心现在需要更多的产品。配送中心 Savant 能够通知商店 Savant，配送车辆已于具体时间出发。在网络结构中处于不同位置的 Savant 软件，对数据具有不同的存储、加工要求。

Savant 的必需标准模块有三个：EMS、RIED、TMS。本说明书着重介绍 Savant 对这三个模块的性能要求、模块的结构和实现细节。其中 EMS 用于读取解读器或传感器中的数据，对数据进行平滑、协同和转发，将处理后的数据写入 RIED 或 database。RIED 是 Savant 特有的一种存储容器，是一个优化的数据库，为了满足 Savant 在逻辑网络中的数据传输速度而设立，它提供与 database 相同的数据接口，但访问速度比 database 快的多。TMS 的功能类似于操作系统的任务管理器，它把由外部应用程序定制的任务转为 Savant 可执行的程序，写入任务进度表，使 Savant 具有多任务执行功能。Savant 支持的任务包括三种类型：一次性任务、循环任务、永久任务。

参考文献

1. **TECHNICAL MANUAL The Savant version 0.1(Alpha)**
   MIT-AUTOID-TM-005 Published February 1,2002. Distribution restricted to Sponsors until May 1,2002.
2. **Amit Goyal Master’s Candidate, MIT lab, TECHNICAL REPORT Savant Guide** MIT-AUTOID-tr0015 Published April,2003
3. **AutoID Savant specification 1.0**
   http://www.epcglobalinc.org/standards_technology/specifications.html